📧 Email Microservice

A simple, lightweight SMTP email sending microservice built with Express.js and Nodemailer. Features automatic sleep/wake functionality to optimize resource usage.

🚀 Features

- ✅ **SMTP Email Sending** - Send emails using any SMTP provider
- ✅ **Flexible Configuration** - SMTP settings provided in request body (no hardcoded secrets)
- ✅ **Multiple Recipients** - Support for sending to multiple email addresses
- ✅ **Rich Content** - Support for both HTML and plain text emails
- ✅ **File Attachments** - Send files as email attachments
- ✅ **Auto Sleep/Wake** - Automatically sleeps after 30 minutes of inactivity
- ✅ **Health Monitoring** - Built-in health check endpoint
- ✅ **CORS Enabled** - Ready for cross-origin requests

📋 API Endpoints

1. Welcome & Usage Instructions

```
GET /
```

Returns welcome message with complete usage instructions and examples.

2. Health Check

```
GET /health
```

Returns service health status, uptime, memory usage, and service state.

3. Send Email

```
POST /send-email
```

Sends an email using SMTP configuration provided in the request body.

4. Wake Up Service

```
POST /wake
```

Wakes up the service if it's sleeping due to inactivity.

5. Manual Sleep (Bonus)

```
POST /sleep
```

Manually puts the service to sleep.

🛠️ Installation & Setup

Installation

```
# Install dependencies
npm install express cors nodemailer
npm install -D @types/express @types/cors @types/nodemailer typescript ts-node
```

Run the Service

```
# Development mode
npx ts-node email-microservice.ts

# Or build and run
```

```
npm run build
npm start
```

📧 Usage Examples

Basic Email Example

```
curl -X POST http://localhost:3001/send-email \
  -H "Content-Type: application/json" \
  -d '{
    "smtpConfig": {
      "host": "smtp.gmail.com",
      "port": 587,
      "secure": false,
      "auth": {
        "user": "your-email@gmail.com",
        "pass": "your-app-password"
      }
    },
    "emailData": {
      "from": "sender@example.com",
      "to": "recipient@example.com",
      "subject": "Test Email from Microservice",
      "text": "Hello! This is a test email from the microservice.",
      "html": "<h1>Hello!</h1><p>This is a <strong>test email</strong> from the
microservice.</p>"
    }
  }'
```

Multiple Recipients Example

```
curl -X POST http://localhost:3001/send-email \
  -H "Content-Type: application/json" \
  -d '{
    "smtpConfig": {
      "host": "smtp.gmail.com",
      "port": 587,
      "secure": false,
      "auth": {
        "user": "your-email@gmail.com",
        "pass": "your-app-password"
      }
    },
    "emailData": {
      "from": "sender@example.com",
```

```
      "to": ["recipient1@example.com", "recipient2@example.com",
"recipient3@example.com"],
      "subject": "Bulk Email Test",
      "html": "<h2>Bulk Email</h2><p>This email was sent to multiple recipients.
</p>"
    }
  }'
```

Email with Attachments Example

```
curl -X POST http://localhost:3001/send-email \
  -H "Content-Type: application/json" \
  -d '{
    "smtpConfig": {
      "host": "smtp.gmail.com",
      "port": 587,
      "secure": false,
      "auth": {
        "user": "your-email@gmail.com",
        "pass": "your-app-password"
      }
    },
    "emailData": {
      "from": "sender@example.com",
      "to": "recipient@example.com",
      "subject": "Email with Attachment",
      "text": "Please find the attachment below.",
      "attachments": [
        {
          "filename": "hello.txt",
          "content": "SGVsbG8gV29ybGQh",
          "encoding": "base64"
        }
      ]
    }
  }'
```

Health Check Example

```
curl http://localhost:3001/health
```

Wake Up Service Example

```
curl -X POST http://localhost:3001/wake
```

🔧 SMTP Provider Examples

Gmail

```
{
  "smtpConfig": {
    "host": "smtp.gmail.com",
    "port": 587,
    "secure": false,
    "auth": {
      "user": "your-email@gmail.com",
      "pass": "your-app-password"
    }
  }
}
```

Outlook/Hotmail

```
{
  "smtpConfig": {
    "host": "smtp-mail.outlook.com",
    "port": 587,
    "secure": false,
    "auth": {
      "user": "your-email@outlook.com",
      "pass": "your-password"
    }
  }
}
```

SendGrid

```
{
  "smtpConfig": {
    "host": "smtp.sendgrid.net",
    "port": 587,
```

```
    "secure": false,
    "auth": {
      "user": "apikey",
      "pass": "your-sendgrid-api-key"
    }
  }
}
```

Mailgun

```
{
  "smtpConfig": {
    "host": "smtp.mailgun.org",
    "port": 587,
    "secure": false,
    "auth": {
      "user": "your-mailgun-smtp-username",
      "pass": "your-mailgun-smtp-password"
    }
  }
}
```

🛡️ Security Features

- **No Hardcoded Secrets**: All SMTP credentials are provided in request body
- **Input Validation**: Comprehensive validation of all input parameters
- **Error Handling**: Proper error handling and informative error messages
- **CORS Protection**: CORS middleware for secure cross-origin requests

🔄 Sleep/Wake Functionality

The microservice automatically:

- **Sleeps** after 30 minutes of inactivity to save resources
- **Tracks** last activity time for all endpoints
- **Provides** wake-up endpoint to reactivate sleeping service
- **Reports** current state in health checks and responses

📊 Response Format

Success Response

```
{
  "success": true,
  "message": "Email sent successfully!",
  "messageId": "message-id-from-smtp",
  "response": "250 OK",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Error Response

```
{
  "success": false,
  "message": "Failed to send email",
  "error": "Error details here",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Health Response

```
{
  "status": "healthy",
  "timestamp": "2024-01-01T12:00:00.000Z",
  "uptime": 3600,
  "serviceState": "active",
  "lastActivity": "2024-01-01T12:00:00.000Z",
  "memory": {
    "rss": 50331648,
    "heapTotal": 20971520,
    "heapUsed": 15728640,
    "external": 1048576
  },
  "version": "1.0.0"
}
```

🚦 Environment Variables

The microservice uses the following optional environment variables:

- `PORT` - Server port (default: 3001)

🐛 Troubleshooting

Common Issues

1. **Gmail Authentication Error**
   - Enable 2-factor authentication
   - Generate an App Password instead of using your regular password
   - Use the App Password in the `pass` field
2. **Service Not Responding**
   - Check if service is sleeping: `GET /health`
   - Wake up the service: `POST /wake`
3. **SMTP Connection Error**
   - Verify SMTP host and port settings
   - Check if your email provider requires specific security settings
   - Ensure credentials are correct

📝 License

ISC License - Feel free to use and modify as needed.

🤝 Contributing

This is a simple microservice template. Feel free to extend it with additional features like:

- Email templates
- Queue management
- Rate limiting
- Logging and monitoring
- Database integration
- Authentication middleware